UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/919,753	08/01/2001	Joseph T. Apuzzo	POU900182US1	6371

46369 7590 03/01/2005

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 COLUMBIA CIRCLE
ALBANY, NY 12203

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 03/01/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/919,753	APUZZO ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 November 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 11/19/2004.

As indicated in Applicant's response, claims 1, 15, 20, 34-35, 40-41, and 51 have been amended. Claims 1-54 are pending in the office action.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claims 15-19, 34 and 40 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

> The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

Claim 15 recites a method for generating test cases with the steps of ascertaining, creating an abstraction matrix, parsing the matrix to automate test cases, separating the test cases based on layers and associating data structures. As such, there is not enough evidence that the steps being performed necessarily require an hardware device to enable the realization of test cases based on analyzing the mapping from an abstract matrix, the test cases being possibly test case on paper without hardware to actually execute them. Hence, the claim fails the practical requirement test for not leading to concrete, tangible,

and useful result. The claim amounts to a mere abstract idea and is rejected for not leading to a statutory subject matter.

Claims 16-19 are rejected for failing to remedy to the deficiencies of claim 15.

Claim 34 recites a system with abstraction matrix and parsing engine and functional verification engine, all of which apparently construed as being software implemented; but does not include sufficient description of hardware support for the same reasons as set forth in claim 15. Hence the claim is rejected for the same reason as claim 15.

Claim 40 recites a medium to store an abstract matrix; and an abstraction engine (not included in the medium) to parse and to generate test cases as in claim 15. Nowhere in the claim is there description of an hardware required to tangibly implement the abstraction engine nor is there a medium for embodying the abstraction engine, which from the specifications (Fig. 3, 4, 5) is a software engine. Absent any tangible device to implement such software engine, the claim amounts to a mere abstract concept of generating test cases without hardware support; and is rejected for being an impractical abstract idea, not leading to concrete, tangible, and useful result, hence to a statutory subject matter.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Passova, USPN: 6,671,874 (hereinafter Passova) in view of Okayasu, USPN: 5,659,554 (hereinafter Okayasu).

As per claim 1, Passova discloses a method of testing a software component comprising:

creating an abstraction matrix by hierarchizing the software component into multiple levels (e.g. *Use Cases* – col. 7, lines 60 to col. 8, line 15; *current level* - col. 4, lines 56-61 – Note: Use cases corresponding to a level of requirements and represented by a model abstracted by matrix structures read on creating an abstraction matrix as recited), the abstraction matrix comprising state and event information taking into account relationships that exist between the multiple levels (e.g. Fig. 1-3; col. 10, lines 20-64; col. 11, lines 20 to col. 12, line 54; col. 15, lines 5-12 - Note: each flow per Use Case represented by Petri model having input/output tuple relationship with alternative flows read on relationships with multiple levels of software Use cases; i.e. the association of matrix to implement the Use Case flow reads on creating a matrix comprising information accounting of relationships between levels or between Petri diagrams),

parsing the abstraction matrix to automatically generate test cases and mapped expected results therefor (e.g. col. 3, lines 61-63; col. 7, lines 9-15; col. 14, lines 8 to col. 15, line 34);

separating the test cases based on the requirement level/Use Cases (*Use Cases* – col. 7, lines 60 to col. 8, line 15) of the software component, and associating data structures (e.g. table 5-6, col. 14; table 7, col. 15) with the separated test cases of the levels, the data structures allowing the test cases of the various levels to be uncorrelated (

Note: input/output to trigger transition response or expected outcomes for one function in a Use Case flow reads on uncorrelated state of input/output with counterparts of another functionality in another instance of flow);

employing the software component in executable form to generate for each level of the software component test case execution from the test cases and mapped expected results for that level; thereby testing the software component (col. 16, line 36 to col. 17, line 30).

But Passova does not specifically teach partitioning of levels as partitioning as layers even though the teaching of Use Cases hint partitioning of the whole product going by level of requirements as mentioned above. Similar to Passova's organizing of software requirements components into a plurality of state diagrams, Okayasu teaches organizing state transition diagrams in hierarchy of diagram representing components, i.e. presented in layers that can act in parallel with each other (Fig. 1-2; Fig 15). Thus, in case Passova's organization of requirement levels in Use cases does not already meet the layering limitation, it would have been obvious for one of ordinary skill in the art at the time the invention was made to layer the software component as taught by Okayasu because this enable better accounting for relationships between requirements when parallel execution has to be performed by the components thus layered, thereby generating tests based on the requirement cases as a whole, thus enhancing reliability (see Okayasu: col. 8, lines 24 to col. 9, line 15).

Nor does Passova explicitly disclose generating test case threads. Passova discloses generating test for events along a dynamically flowing path representing requirements Use Case. The concept of events testing is evidenced in Okayasu's

development using simultaneous execution of layered states in the transition diagram (col. 5, line 51 to col. 6, line 14) such that the concept of creating threads to concurrently execute is implied. Therefore if Passova does not have test cases as threads, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the test code as threads thus suggested by Okayasu in order to impart the dynamic implementation to response required from the event/transitions flow as mentioned above.

Further, Passova does not disclose executing in parallel at least some of the test case execution threads for at least one layer of the software component; but because of the input/output mapping as mentioned above, the concept of threads being invoked in testing of actions called for in a Petri flow has been addressed above. The limitation that threads are parallel is either implicit or obvious by virtue of the teaching by Okayasu that layers of software can be tested in parallel to yield reliable results as desired by Passova (col. 15, lines 9-12) and Okayasu (col. 8, lines 24 to col. 9, line 15)

As per claim 2, Passova discloses creating the abstraction matrix from a functional specification of the software component (e.g. col. 4, lines 17-29).

As per claim 3, Passova teaches that the parsing, the separating, the employing and the executing comprise automated processes (col. 3, line 61 to col. 4, line 55).

As per claim 4, Passova in view of Okayasu teaches executing in parallel at least some of the test case execution threads for multiple layers of the software component (refer to rationale of corresponding rejection in claim 1).

As per claim 5, Passova teaches executing threads for steps in a flow in a Use case; and in combination with Okayasu discloses executing in parallel at least some test case execution threads of each layer of the software component.

As per claim 6, Passova (in view of Okayasu) discloses separating the test cases based on layers of the software component results in each layer of the software component having multiple test cases (see Fig. 1-3) associated therewith.

As per claim 7, Passova (in view of Okayasu) discloses creating the abstraction matrix so that the abstraction matrix describes each layer of the software component (e.g. Fig. 1-3; col. 10, lines 20-64; col. 11, lines 20 to col. 12, line 54; col. 15, lines 5-12).

As per claim 8, Passova discloses that creating comprises creating a separate abstraction file for each layer of the layers of the software component (e.g. Fig. 7).

As per claim 9, Passova discloses providing attributes (*annotations, column, position, variable, row, intersection* - col. 9, line 32 to col. 10, line 64) for use in generating the test cases and mapped expected results therefore.

As per claims 10 and11, Passova discloses creating the abstraction matrix to comprise state information and events that are associated with each state of the software component (e.g. col. 11, line 20 to col. 12, line 54), including current state information and next state information, and wherein for at least one current state the event information comprises an event which leads from that current state to go least one next state (Fig. 1-6).

As per claim 12, Passova does not specifically teach taht the software component comprises an operating system component; but in view of the compatibility of Passova's

product between different operating systems (e.g. col. 3, lines 23-32), the comprising of an OS component is implicitly disclosed.

As per claim 13, Passova discloses creating the abstraction matrix to identify a minimal number of states (e.g. basic flow – col. 9, lines 36-52) required to model the software component.

As per claim 14, Passova discloses associating data structures for the separated test cases of the layers comprises associating information on the relationships between layers (table 5-6, col. 14; table 7, col. 15--Note: input/output to trigger transition response or expected outcomes for one function in a Use Case flow reads on uncorrelated state of input/output with counterparts of another functionality in another instance of flow), thus allowing test cases of the various layers to be uncorrelated.

As per claim 15, Passova discloses a method of generating test cases for use in creating a software component, said method comprising:

ascertaining a functional specification of the software component (col. 3, lines 21-37; col. 4, lines 17-29);

creating (an abstraction matrix);

parsing (abstraction matrix); and

separating (the test cases ... data structures); just as recited in claim 1.

These limitations will be rejected with the same rationale as set forth in claim 1 correspondingly, including the rationale corresponding to the partitioning by layer limitation using Okayasu.

As per claims 16-19, these claims correspond to claims 3, 8, 11, 14, respectively; hence are rejected with the corresponding rejection as set forth therein.

As per claim 20, this claim is a system means-plus-function corresponding to claim 1; hence is rejected with the corresponding rejection as set forth therein.

As per claims 21-33, these claims correspond to claims 2-14, respectively; hence are rejected with the corresponding rejection as set forth therein.

As per claim 34, Passova discloses a system for testing a software component, comprising:

an abstraction matrix that describes the software component, the abstraction matrix partitioning the software component into multiple levels and comprising state and event information (*Use Cases* – col. 7, lines 60 to col. 8, line 15; *current level* - col. 4, lines 56-61; Fig. 1-3; col. 10, lines 20-64; col. 11, lines 20 to col. 12, line 54; col. 15, lines 5-12) taking into account relationships that exist between the multiple levels;

an abstraction engine for parsing the abstraction matrix and automatically generating test cases and mapped expected results therefor (e.g. col. 3, lines 61-63; col. 7, lines 9-15; col. 14, lines 8 to col. 15, line 34);

wherein the abstraction engine separates the test cases based on the level of the software component, and associates data structures (*Use Cases* – col. 7, lines 60 to col. 8, line 15; table 5-6, col. 14; table 7, col. 15) with the separate test cases of the level, the data structures allowing the test cases of the various levels to be uncorrelated; and

a functional verification test engine (e.g. col. 3, lines 47-60) adapted to take the software component in executable form and generate for each level of the software component test case execution from the test cases and mapped expected results for that level; thereby testing the software component (col. 16, line 36 to col. 17, line 30).

But Passova does not specifically teach partitioning of levels as partitioning as layers; nor does Passova explicitly disclose generating test case threads; nor disclose executing in parallel at least some of the test case execution threads for at least one layer of the software component. But these limitations have been addressed in claim 1 using Okayasu and the corresponding rationale.

As per claim 35, Passova discloses a system for generating test cases for use in testing a software component, said system comprising the same limitations such as:

a functional specification...;

an abstraction matrix...;

means for parsing...; and

means for separating ... as recited in claim 15.

Hence these limitations will be rejected with the same rationale as set forth in claim 15 correspondingly.

As per claims 36-39, these claims correspond to claims 3, 8, 11, 14, respectively; hence are rejected with the corresponding rejection as set forth therein.

As per claim 40, Passova discloses a system for generating test cases for use in testing a software component, the system comprising:

a storage medium for storing an abstraction matrix that describes the software component by partitioning the software component into multiple levels, the abstraction matrix, which comprises state and event information taking into account relationships that exist between the multiple levels (*Use Cases* – col. 7, lines 60 to col. 8, line 15; *current level* - col. 4, lines 56-61; Fig. 1-3; col. 10, lines 20-64; col. 11, lines 20 to col.

12, line 54; col. 15, lines 5-12), having been created from a functional specification of the software component (e.g. col. 4, lines 17-29);

an abstraction engine for automatically retrieving and parsing the abstraction matrix to automatically generate test cases and mapped expected results therefor (e.g. col. 3, lines 61-63; col. 7, lines 9-15; col. 14, lines 8 to col. 15, line 34); and

wherein the abstraction engine separates the test cases based on the levels of the software component and associates data structures with the separated test cases of the levels, the data structures allowing the test cases of the various levels to be uncorrelated (*Use Cases* – col. 7, lines 60 to col. 8, line 15; table 5-6, col. 14; table 7, col. 15).

But Passova does not specifically teach partitioning of levels as partitioning as layers. But these limitations have been addressed in claim 1 using Okayasu and the corresponding rationale.

As per claim 41, Passova discloses a least one program storage device readable by machine, tangibly embodying at least one program of instructions executable by the machine to form a method for testing a software component, comprising:

storing an abstraction matrix (... by partitioning the software component into multiple levels);

parsing the abstraction matrix (.. generate test cases);

separating the test cases (... associating data structures ... various levels to be uncorrelated);

employing (... in executable form to generate ... test case execution ... for that level); and

executing (some of the test case ... testing the software component).

All of these limitations have been addressed in the corresponding rejections in claim 1, respectively.

However, Passova does not specifically teach partitioning of levels as partitioning as layers; nor does Passova explicitly disclose generating test case threads; nor disclose executing in parallel at least some of the test case execution threads for at least one layer of the software component. But these limitations have been addressed in claim 1 using Okayasu and the corresponding rationale.

As per claims 42-50, these claims correspond to claims 4-8, 10, 11, 12, and 14, respectively; hence are rejected with the corresponding rejection as set forth therein.

As per claim 51, Passova discloses at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method for generating test cases for use in testing a software component, the method comprising:

storing an abstraction matrix (... by partitioning the software component into multiple levels);

parsing the abstraction matrix (.. generate test cases);

separating the test cases (... associating data structures ... various levels to be uncorrelated).

These limitations will be rejected with the same rationale as set forth in claim 41 correspondingly using the rejection of claim 1, including the rationale corresponding to the partitioning by layer limitation using Okayasu.

As per claims 52-54, these claims correspond to claims 8, 11, 14, respectively; hence are rejected with the corresponding rejection as set forth therein.

Response to Arguments

6. Applicant's arguments with respect to claims 1-54 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
February 8, 2005

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100